Лабораторна робота №4. Перетворення віртуальних адрес

Мета лабораторної роботи наступна:

- ознайомитись з елементами рівня архітектури операційної системи на прикладі функції реалізації і підтримки віртуальної пам’яті;
- навчитись перетворювати віртуальні адреси у фізичні.

1. Вимоги до апаратного і програмного забезпечення

Для виконання лабораторної роботи діють наступні вимоги щодо програмного та апаратного забезпечення:

1) компілятор мови C — gcc версії 14.2 або вище
2) компілятор мови C++ — g++ версії 14.2 або вище.

2. Загальне завдання та термін виконання

Завдання лабораторної роботи наступне: розширити реалізовану у лабораторній роботі №3 програму мовою C або C++, таким чином, щоб крім зчитування послідовності команд (програми) з файлу, програма також виконувала заміну віртуальних адрес на фізичні в командах, що визначаються варіантом. Заміна відбувається в залежності від типу організації віртуальної пам’яті, який також визначається варіантом. Заміна адреси відбувається у випадку, якщо сторінка та/або сегмент знаходиться в оперативній пам’яті (ОП). Якщо потрібна віртуальна сторінка та/або сегмент відсутні в ОП, тоді має бути виведене повідомлення про помилку відсутності сторінки/сегменту, й аналіз команд має бути продовжено. Таблиця сторінок/сегментів задається у файлі формату CSV.

Розширення програми з л.р. №3 складатиметься з наступних компонентів:

1. Модуль зчитування таблиці сторінок/сегментів з файлу CSV і створення внутрішнього представлення відповідної таблиці (або таблиць);
2. Змінений модуль з зчитування і аналізу команд з текстового файлу, що також виконує заміну віртуальних адрес в зчитаних командах на фізичні;
3. Змінений/розширений модуль тестування, що містить тестові утиліти і тести реалізованої програми.

В результаті виконання лабораторної роботи має бути підготовлено **звіт**, який містить:

1) титульний аркуш;
2) загальне завдання лабораторної роботи;

3) варіант і завдання за варіантом (інформація про варіанти наведена в п. 4);
4) лістинг реалізованої програми (усіх модулів);
5) результати тестування програми для декількох наборів тестових даних. Тести мають виконувати перевірку всіх пунктів основного завдання, завдання за варіантом і продемонструвати коректність роботи реалізованої програми.

Граничний **термін виконання** лабораторної роботи визначається викладачем в інформаційному листі, що надсилається на пошту групи та/або в групу в телеграмі разом із даними методичними вказівками. В разі недотримання граничного терміну виконання лабораторної роботи, максимальна оцінка за роботу буде знижена на 1 бал за кожен тиждень протермінування. Кількість балів за дотримання дедлайну даної лабораторної роботи визначається силабусом (РСО). Ця ж кількість відповідає максимальній кількості балів, на яку може бути знижена оцінка за лабораторну роботу в разі недотримання дедлайну.

3. Методичні вказівки

В даному розділі наведено методичні вказівки щодо виконання пунктів завдання лабораторної роботи.

3.1 Типи організації пам’яті

В залежності від варіанту необхідно реалізувати підтримку одного з наступних *типів організації пам’яті*:

1) сторінкова;
2) сегментна;
3) сегментно-сторінкова.

Основний теоретичний матеріал, на якому базується дана лабораторна робота, викладено у відповідній лекції курсу «Архітектура комп’ютерів. Архітектура для програмістів».

3.1.1 Сторінкова організація пам’яті

При *сторінковій організації пам’яті* віртуальний адресний простір ділиться на сторінки — фрагменти адресного простору фіксованої довжини. Оскільки віртуальний адресний простір зазвичай є значно більшим за розмір основної пам’яті, необхідні сторінки завантажуються в основну пам’ять тоді, коли в них виникає необхідність (відбувається перехід до виконання інструкції на деякій сторінці, або відбувається звернення до даних на цій сторінці). При цьому код програми оперує віртуальними адресами. Для того, щоб отримати фізичну адресу — адресу в основній пам’яті, за якою знаходиться адресована команда або дані, — використовується *таблиця сторінок*, що містить інформацію про те, чи присутня певна віртуальна сторінка в основній пам’яті, і якщо присутня, то який *сторінковий кадр* їй відповідає (тобто, де знаходиться ця віртуальна сторінка в основній пам’яті).

На рис. 4.1 наведено приклад отримання фізичної адреси для деякої віртуальної адреси. При цьому використовується 32-розрядна адреса, що складається з 20 розрядів номеру віртуальної сторінки і 12 розрядів зміщення в межах цієї сторінки. Тобто, розмір сторінки складає 4 Кбайт. Припустимо, що розмір ОП складає 16 Кбайт. В такому разі, розмір фізичної адреси складає 14 бітів (2 біти — номер сторінкового кадру і 12 бітів — зміщення в сторінці). Варто зауважити, що розмір сторінки може відрізнятись, відповідно й розрядність поля номеру сторінки у 32-х бітах віртуальної адреси може відрізнятись.

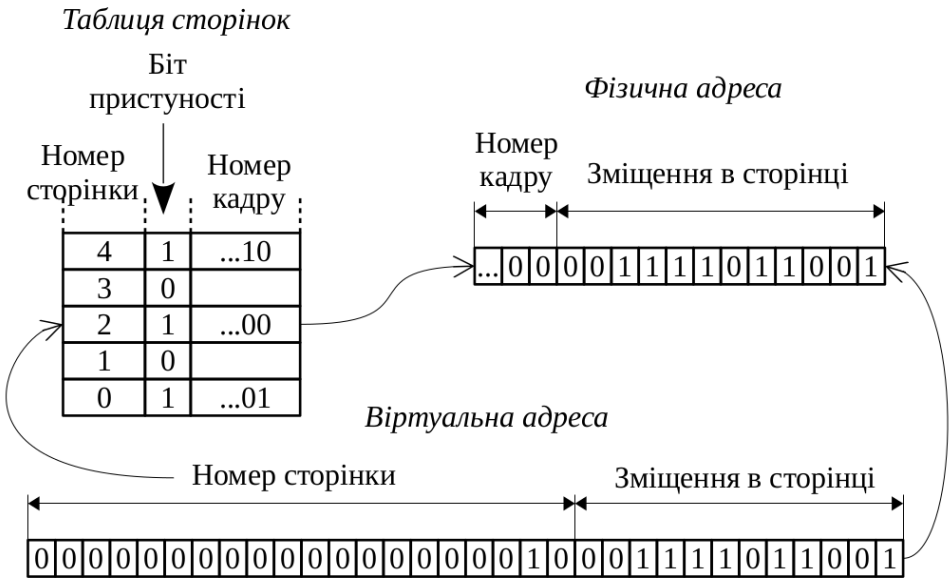

Рисунок 4.1 — Перетворення віртуальної адреси у фізичну при сторінковій організації пам’яті

Структура файлу CSV таблиці сторінок наступна: кожен рядок містить номер сторінки, прапорець присутності сторінки в основній пам’яті і номер сторінкового кадру віртуальної сторінки.

Якщо під час перетворення віртуальної адреси виявиться, що віртуальна сторінка відсутня в основній пам’яті — необхідно вивести в термінал повідомлення про помилку відсутності сторінки, вказавши всю команду і конкретну адресу, для якої виникла помилка.

Якщо під час перетворення віртуальної адреси виявиться, що вказаної в адресі віртуальної сторінки не існує — також має бути виведене відповідне повідомлення.

3.1.2 Сегментна організація пам’яті

Сегмент — це окремий адресний простір, що містить деяку структуру, процедуру, масив або інший об’єкт. Сегментація пам'яті дає змогу зображати віртуальний адресний простір як сукупність незалежних блоків змінної довжини (сегментами). Програма може складатись з кількох сегментів. Кожен сегмент описується в дескрипторі сегменту, що зберігається в *таблиці дескрипторів сегментів*. Віртуальні адреси при сегментній організації складаються з номеру сегмента (номеру дескриптора сегменту в таблиці) та зміщення в цьому сегменті (рис. 4.2).

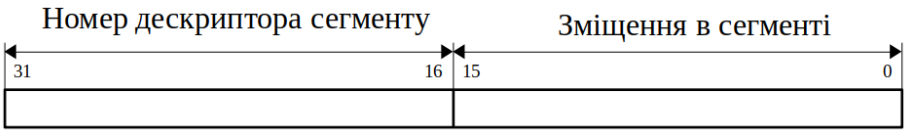

Рисунок 4.2 — Структура віртуальної адреси при сегментній організації пам’яті

Варто зауважити, що розрядність поля номера дескриптора може бути різною в межах 32-х розрядів віртуальної адреси.

Для перетворення віртуальної адреси у фізичну, необхідно взяти номер дескриптора сегменту з віртуальної адреси, отримати відповідний дескриптор (рис. 4.3). Дескриптор складається з базової адреси, поля розміру сегменту (в байтах) і прапорців, до числа яких входить прапорець присутності в основній пам’яті. З дескриптора сегменту необхідно отримати інформацію про *базову адресу* (початкову адресу) сегменту в основній пам’яті. Тоді додати до базової адреси зміщення в сегменті з віртуальної адреси і отримати фізичну адресу (рис. 4.4).

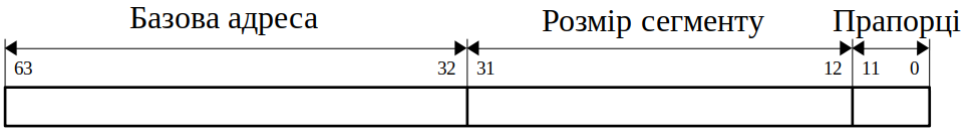

Рисунок 4.3 — Дескриптор сегмента

Структура файлу CSV таблиці дескрипторів сегментів сторінок наступна: кожен рядок містить базову адресу, розмір сегмента (в байтах) і прапорець присутності сегменту в основній пам’яті.

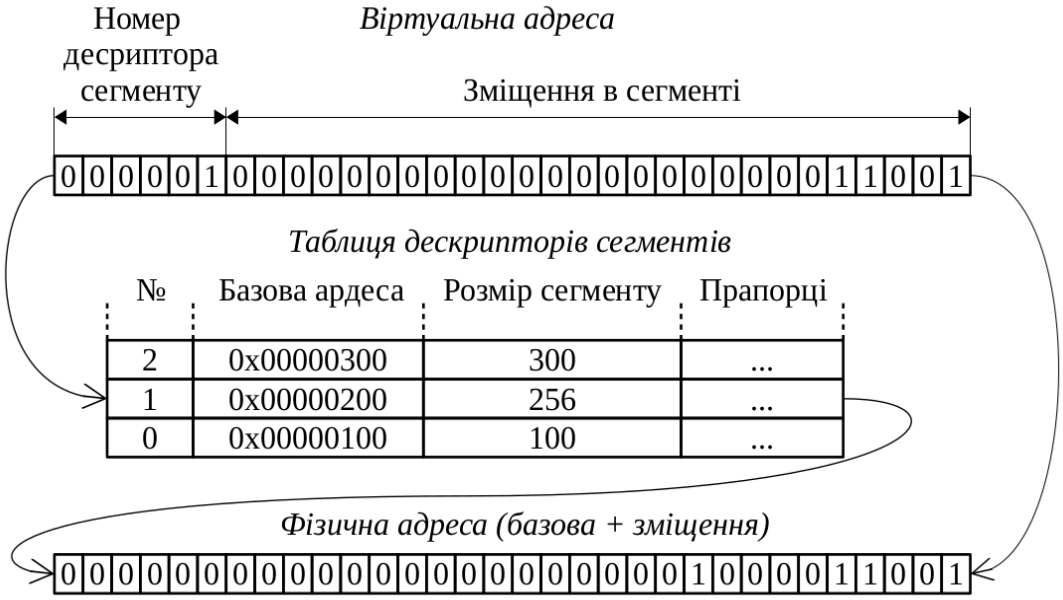

Рисунок 4.4 — Перетворення віртуальної адреси у фізичну при сегментній організації пам’яті

Якщо під час перетворення віртуальної адреси виявиться, що сегмент відсутній в основній пам’яті — необхідно вивести в термінал повідомлення про помилку відсутності сегменту, вказавши всю команду і конкретну адресу, для якої виникла помилка.

Якщо під час перетворення віртуальної адреси виявиться, що вказаного в адресі сегменту не існує, або якщо зміщення в сегменті перевищує розмір сегменту — також має бути виведене відповідне повідомлення.

3.1.3 Сегментно-сторінкова організація пам’яті

При *сегментно-сторінковій організації пам’яті* програма ділиться на сегменти, а сегменти — на сторінки фіксованої довжини. Відповідно віртуальна адреса в адресному просторі програми складається з ідентифікатора сегменту, номера віртуальної сторінки та зміщення в межах сторінки (рис. 4.5). Варто зауважити, що розрядність номера дескриптора та розмір сторінки можуть бути різними в межах 32-х розрядів віртуальної адреси (задаються варіантом).

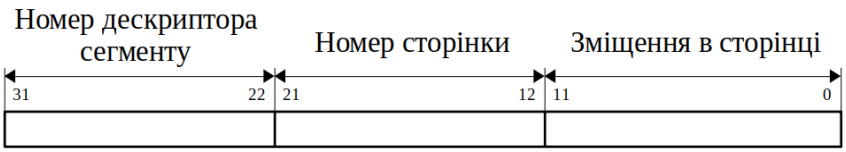

Рисунок 4.5 — Структура віртуальної адреси при сегментно-сторінковій організації пам’яті

На відміну від сегментної організації, дескриптор сегментів при сегментно-сторінковій організацій замість базової адреси сегменту містить адресу таблиці сторінок, кількість сторінок сегменту і набір прапорців (рис. 4.6). *Будемо вважати, що таблиця сторінок зберігається не в основній пам’яті, а в окремій, тож при реалізації посилання на таблицю мовою С замість адреси варто використати посилання на об’єкт таблиці.*

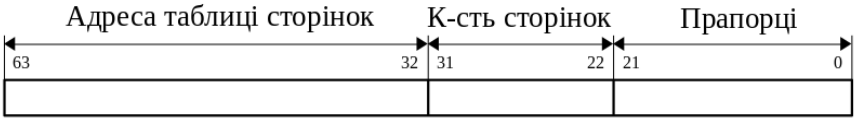

Рисунок 4.6 — Дескриптор сегменту при сегментно-сторінковій організації пам’яті

Для перетворення віртуальної адреси у фізичну, необхідно взяти номер дескриптора сегменту з віртуальної адреси, отримати відповідний дескриптор. З дескриптора сегменту отримується адреса таблиці сторінок сегменту. З таблиці сторінок отримується номер сторінкового кадру. Номер кадру об’єднується зі зміщенням у сторінці й отримується фізична адреса (рис. 4.7).

Структура файлу CSV таблиці дескрипторів сегментів сторінок наступна: кожен рядок містить шлях до файлу таблиці сторінок і кількість сторінок сегмента (прапорці відсутні).

Структура файлу CSV таблиці сторінок така сама, як при сторінковій організації пам’яті.

Якщо під час перетворення віртуальної адреси виявиться, що сегмент відсутній в таблиці дескрипторів — необхідно вивести в термінал повідомлення про помилку відсутності сегменту, вказавши всю команду і конкретну адресу, для якої виникла помилка. Аналогічно, необхідно повідомити про помилку відсутності сторінки в разі відсутності сторінки в основній пам’яті.

Якщо номер сторінки перевищує розмір сегменту (кількість його сторінок) — також має бути виведене відповідне повідомлення.

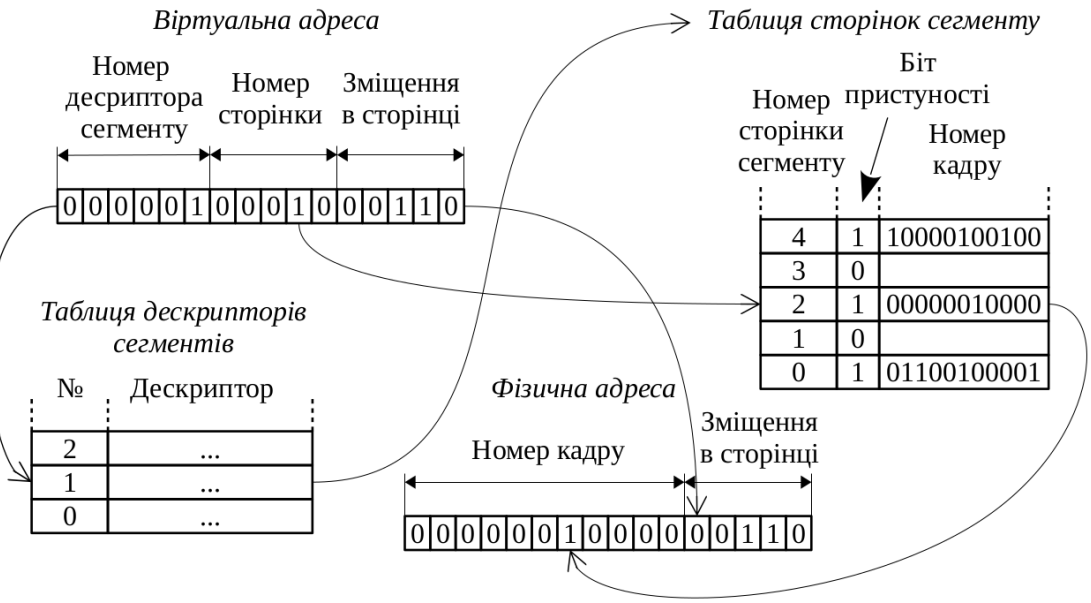

Рисунок 4.7 — Перетворення віртуальної адреси у фізичну при сегментно-сторінковій організації пам’яті

3.2 Вхідні дані

На вхід програми подаються наступні файли:

- Файли в форматі CSV з таблицями сегментів та/або сторінок — *файли таблиць пам’яті* (для деяких варіантів це може бути лише один файл). Формат файлів визначено в п. 3.1 для кожного виду організації пам’яті окремо;
- Текстовий файл, що містить деяку послідовність команд (програму) — *файл програми*.

3.3 Модуль зчитування файлів таблиць пам’яті

Даний модуль має містити функції для розбору файлів таблиць і створення внутрішнього представлення таблиць. Для реалізації внутрішнього представлення можна використовувати структури. Можливим варіантом інтерфейсних функцій та структур може бути (але не обов’язково) наступний:

```
// types
struct segment_table {
  segment_table_record* record;
  int records_number;
}

struct segment_table_record {
  ...
}

// functions
segment_table parse_segment_table(char* filename);
pages_table parse_pages_table(char* filename);
// ... and so on
```

Варто зауважити, що краще розбивати код на менші функції, кожна з яких виконує окрему логіку, замість того, щоб реалізовувати щось складне в одній функції. Наприклад, розбір рядка в файлі можна винести в окрему функцію.

3.4 Модуль зчитування файлу програми, аналізу команд і заміни віртуальних адрес

Даний модуль має виконувати ті ж функції, що навдені в п. 3.4 л.р. №3. Крім того:

1. Якщо в якійсь з команд зустрічається віртуальна адреса (один з операндів команд), вона має бути проаналізована й для неї має бути визначена фізична адреса (див. п. 3.4.1, 3.4.2). При цьому використовуються таблиці сторінок та/або сегментів, зчитані з файлів таблиць пам’яті (п. 3.3).

3.4.1 Перетворення віртуальних адрес у фізичні

Переведення віртуальної адреси у фізичну відбувається в залежності від організації пом’яті (сторінкової, сегментної, сегментно-сторінкової). Нехай маємо сегментну організацію, й наступний вміст таблиці сегментів (таблиця 4.1).

Таблиця 4.1 — Вміст таблиці сегментів

№	Базова адреса	Розмір (байт)	Прапорець присутності в пам’яті
0		100	0
1	0x00001A00	200	1
2		500	0
3	0x00000100	1000	1

Нехай розмір поля номера дескриптора у віртуальній адресі складає 10 бітів (старші біти адреси), тоді решта 22 біти (молодші біти) — зміщення в сегменті. Наведемо кілька варіантів переведення віртуальної адреси у фізичну:

- 0x00C000A1 — адреса складається з номера дескриптора 00 00 00 00 11 (3 в десятковій системі) і зміщення в сегменті 00 00 00 00 00 00 00 10 10 00 01 (або 161 в десятковій системі). Таку адресу перевести можна, оскільки сегмент знаходиться в пам’яті і зміщення валідне. Фізична адреса для цієї віртуальної — 0x000001A1;
- 0x00C004D0 — адреса складається з номера дескриптора 00 00 00 00 11 (3 в десятковій системі) і зміщення в сегменті 00 00 00 00 00 01 00 11 01 00 00 (або 1232 в десятковій системі). Така адреса некоректна, оскільки зміщення в сегменті перевищує розмір сегменту. Тому для адрес необхідно вивести повідомлення про помилку;
- 0x008000D0 — адреса складається з номера дескриптора 00 00 00 00 10 (2 в десятковій системі) і зміщення в сегменті 00 00 00 00 00 00 00 11 01 00 00 (або 208 в десятковій

системі). Таку адресу перевести можна, оскільки сегмент існує і зміщення валідне. Проте сам сегмент відсутній в ОП, тому необхідно повідомити про помилку відсутності сегменту;

- помилка також буде у випадку, якщо віртуальна адреса міститиме номер дескриптора сегменту, якого нема в таблиці сегментів.

Аналогічним чином відбувається перетворення і вивід повідомлень про помилки і при реалізації інших типів організації пам’яті.

3.4.2 Приклад роботи

Вивід в термінал під час аналізу файлу програми, наведеної в п. 3.4.1 л.р. №3, і за організації пам’яті, будови віртуальної адреси та вмісту таблиці сегментів, що наведені в п. 3.4.1, буде наступний:

```
1C 00 02:
MOV  R0,  2

1B 01 00 C0 00 A1:
MOV  R1,  [0x000001A1]

01 01:
ADD  R0, R1

91 FF FF FF 00:
ERROR: failed to convert virtual address [0xFFFFFF00] because segment 1023 doesn’t exist
JMP  [0xFFFFFF00]
```

Адреси виводяться в квадратних дужках в шістнадцятковому вигляді з префіксом 0x. Константи і зміщення — десятковими знаковими числами. Регістри — за ім’ям, що є поєднанням літери R і номеру регістру в десятковому вигляді.

4. Варіанти завдань

Номер варіанту обирається згідно зі списком групи з варіантами, який був надісланий на пошти груп на початку семестру. Варіант береться по модулю 15 — якщо варіант 16, тоді виконується варіант 1 з таблиці 4.2, якщо 17 — тоді 2 і т. д.

Для кожного варіанту визначається тип організації пам’яті та параметри віртуальних адрес. Список команд — такий симий, як в л.р. №3.

Параметри віртуальних адрес наступні:

- розмір сторінки (РС, байт) — дозволяє визначити розрядність зміщення в сторінці;
- розмір таблиці дескрипторів сегментів (РТД, од.) — дозволяє визначити розрядність поля номера дескриптора сегмента;
- розрядність поля номера сторінки визначається як результат віднімання РС і РТД від 32-х.

Тестові набори необхідно створити самостійно.

Таблиця 4.2 — Варіанти завдань

Варіант	Тип організації пам’яті	Параметри віртуальної адреси
1	сторінкова	РС: 4 Кбайт
2	сегментна	РТД: 1024
3	сегментно-сторінкова	РС: 4 Кбайт, РТД: 1024
4	сторінкова	РС: 2 Кбайт
5	сегментна	РТД: 512
6	сегментно-сторінкова	РС: 1 Кбайт, РТД: 4096
7	сторінкова	РС: 1 Кбайт
8	сегментна	РТД: 2048
9	сегментно-сторінкова	РС: 8 Кбайт, РТД: 512
10	сторінкова	РС: 8 Кбайт
11	сегментна	РТД: 256
12	сегментно-сторінкова	РС: 2 Кбайт, РТД: 2048
13	сторінкова	РС: 512 байт
14	сегментна	РТД: 4096
15	сегментно-сторінкова	РС: 512 байт, РТД: 8192

5. Контрольні питання

1) Що таке віртуальна пам’ять?
2) Що таке сторінкова організація пам’яті?
3) Що таке сторінка, сегмент?
4) Як реалізується сегментно-сторінкова організація пам’яті?
5) Що таке внутрішня і зовнішня фрагментація?
6) Які є політики дефрагментації?